International Journal of Computer Information Systems and Industrial Management Applications
ISSN 2150-7988 Volume 18(13s) 2026 pp. 102-117
https://doi.org/10.70917/ijcisim-2026-4036
Article

Multi-Channel Deep Learning Filtering Method for Accurate Heart Rate Detection and Performance Comparison with Existing Technique

Jyoti Vitthal Chhatrband[1], Rekha Chaudhary[2], Parulpreet Singh[3]

[1]School of Electronics and Electrical Engineering, Lovely Professional University, Phagwara, Punjab 144411, India.
Email: jojitagurav@gmail.com
[2]School of Electronics and Electrical Engineering, Lovely Professional University, Phagwara, Punjab 144411, India.
Email: rekha.30912@lpu.co.in
[3]School of Electrical and Electronics Engineering, Lovely Professional University, Phagwara, Punjab 144411, India.
Email: parulpreet.23367@lpu.co.in

Abstract: A multi-channel deep learning filtering method is suggested in this study to make heart rate tracking more accurate. This method uses strong deep learning frameworks. Advanced signal processing methods, such as noise removal, butter bandpass filtering, and peak recognition algorithms, were used on the UBFC-RPPG dataset to get ground truth heart rate signals. After that, frames from the movies in the collection were taken out, resized to standard sizes, and face regions of interest (ROIs) like the left cheek, right cheek, and temples were found. These ROIs were used to figure out heartbeat signals over time. These signals were then sent to mixed deep learning models. Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) networks are combined in the deep learning design to make a mixed system for efficiently extracting time features. As shown by the results, this combination model works better than either LSTM or GRU models used alone. The suggested method is very good at finding heart rate signs, as shown by loss metrics and accuracy graphs. A comparison with other methods shows that this one works better on standard datasets in terms of accuracy and dependability. This multi-channel deep learning filtering method is a big step forward in heart rate recognition, especially when it comes to using deep learning and face video processing to get better results. The method could be used in healthcare tracking systems because it gives a safe and effective way to estimate heart rate without touching the person.

Keywords: Heart Rate Detection, Multi-Channel Filtering, Deep Learning, Facial Video Processing, LSTM-GRU Hybrid Model, Signal Processing

1. Introduction

Heart rate tracking that works correctly is a key part of health and fitness monitoring tools. As non-invasive techniques become more popular, strong and accurate ways to check heart rate without touching the person directly are needed more and more. Video-based heart rate tracking methods, which use photoplethysmography (PPG) data taken from movies of people's faces, have become a potential option to traditional monitors that people wear. These methods are convenient and easy to use, but they have problems like noise interference, changes in lighting, and motion flaws that can make them less accurate. To solve these problems, we suggest a new multi-channel deep learning filtering method that will make heart rate recognition more accurate. The main idea behind this think about is to induce way better comes about by combining progressed profound learning strategies with flag preparing instruments. We get PPG signals and real heart rate information from the UBFC-RPPG dataset, which may be a common collection for evaluating heart rate. This will be utilized as a standard to judge how well the suggested strategy works. The

method incorporates a full workflow that begins with taking care of the ground truth flag, which incorporates getting freed of commotion, finding crests, and calculating the heart rate accurately. This makes sure that the data is clean and precise for the following video-based ponder. Video examination of faces is at the heart of our method. The movies are broken up into outlines at standard times, adjusted to form beyond any doubt everything is the same estimate, and after that inspected to find areas of intrigued (ROIs) just like the confront, left cheek, and right cheek. These ROIs are exceptionally vital for picking up on little changes in skin tone caused by blood stream, which appear how quick your heart is beating. A blended profound learning demonstrate with Long Short-Term Memory (LSTM) and Gated Repetitive Unit (GRU) systems is utilized to prepare the highlights that were taken from these parts of the confront. This combined model takes advantage of the finest highlights of both plans. LSTM is extraordinary at recognizing long-term time connections, and GRU is great at making computations speedier.

The new idea behind the recommended strategy is its multi-channel structure, which analyzes information from diverse parts of the face at the same time to make it more exact and steady. By utilizing information from distinctive parts of that confront, this method reduces the impact of commotion and movement artifacts. The blended LSTM-GRU demonstrate progresses performance even more by finding complex patterns within the data's time arrangement. Testing the recommended show against current strategies appears that it works way better, with higher accuracy and lower mistake rates compared to the current methods. Our study adds to the growing amount of work on video-based heart rate tracking by fixing some of the most important problems with current methods. Using advanced deep learning methods along with multi-channel data processing is a useful option for real-life situations. In fields like telemedicine, fitness tracking, and stress recognition, where reliable and painless heart rate recording is needed, this method could be useful. This study makes progress in the field of heart rate monitoring by using cutting edge deep learning methods and focused on a multi-channel approach. It not only makes tracking much more accurate, but it also sets up a system that can be used in a wide range of healthcare and exercise settings.

2. Related Work

In order to get more exact heart rate readings, non-invasive methods have come a long way, especially video-based photoplethysmography (PPG) methods. These methods use changes in the skin tone of the face caused by blood flow to figure out heart rate. They are easier to use than traditional methods that rely on touch. Existing techniques have a lot of promise, but they have problems with noise interference, motion flaws, and changing environments, so strong answers are needed. Early attempts to find heart rates in videos used signal decomposition methods, like Independent Component Analysis (ICA) and Principal Component Analysis (PCA), to separate PPG signals from movies of faces [4, 5]. These methods worked well to get rid of noise in controlled settings, but they didn't work as well in places with a lot of movement. Bandpass filters and other types of signal filtering have been used in recent studies to reduce noise and improve signal quality [6, 7]. But these approaches are often limited because they can't handle the difficulties of the real world. Deep learning has changed the way heart rate tracking is done by using its strong feature extraction tools. A lot of people use Convolutional Neural Networks (CNNs) to process video data of faces and pull out traits that show blood flow [8, 9]. CNNs are great at recording location information, but they can't model time relationships, which is needed to get a good estimate of heart rate. To fix this problem, Recurrent Neural Networks (RNNs), especially Long Short-Term Memory (LSTM) networks, have been created to find trends in PPG data over time [10, 11]. Multiple channels of information from different parts of the face are combined to give a full picture of bodily cues. Recently, multi-channel processing and deep learning have been used together to get the best heart rate estimate results [15], [17].

When compared to other methods, it is clear that the suggested multi-channel deep learning screening method works well. Traditional methods [16], like ICA and PCA, have trouble staying accurate when things are changing, especially when there are a lot of motion artifacts. Bandpass filtering and other signal processing methods make small steps forward but can't fix the problems with single-channel systems [20]. Deep learning-based methods are very useful, but they don't always have the durability needed for real-world use. The suggested method gets much better accuracy and stability by combining multi-channel processing with a mixed LSTM-GRU design [18], [19]. This study shows how this works. Video-based methods for detecting heart rates have been used in many fields, such as exercise, healthcare, and stress tracking. It would be hugely helpful for telemedicine and smart tech to be able to correctly measure heart rate in non-invasive and remote settings [21, 22]. However, these methods need to be used in the real world, which means solving problems like noise, changing lights, and making the computers work faster.

Table 1: Summary of related work

Method	Algorithm	Key Finding	Details	Limitation	Scope

Signal Decomposition [7]	ICA	Extracts PPG signals effectively in controlled settings	Isolates independent components from mixed signals	Poor performance under motion artifacts	Suitable for static conditions
Signal Decomposition [9]	PCA	Reduces noise in facial PPG signals	Projects data onto principal components for dimensionality reduction	Loses subtle signal variations	Enhances preprocessing in combination with other methods
Signal Filtering [11]	Butterworth Bandpass Filter	Removes high-frequency noise in PPG signals	Smoothens the signal for improved peak detection	Ineffective under complex noise environments	Useful for preprocessing in structured environments
Adaptive Filtering [12]	Kalman Filter	Tracks dynamic signal changes in real-time	Dynamically adjusts filter parameters based on signal characteristics	High computational cost	Real-time applications requiring adaptive adjustments
Video Processing [13]	Optical Flow	Tracks pixel intensity changes for motion compensation	Computes motion vectors to reduce artifact impact	Sensitive to lighting changes	Effective for video-based physiological signal extraction
Signal Recovery [14]	Wavelet Transform	Recovers weak PPG signals	Decomposes signals into frequency components for denoising	Complex implementation	Suitable for signals with non-stationary noise
ROI-Based Processing [15]	Face Detection (Haar)	Identifies facial regions for PPG extraction	Detects key facial features (cheeks, forehead)	Low accuracy in non-frontal faces	Improves preprocessing for multi-channel data
ROI-Based Processing [16]	CNN-Based Face Detection	Enhanced ROI extraction accuracy	Uses deep learning for precise facial feature localization	Computationally intensive	Applicable in high-accuracy healthcare monitoring
Motion Artifact Removal [17]	Adaptive Thresholding	Reduces motion artifacts effectively	Adjusts thresholds dynamically to remove spurious peaks	Sensitive to extreme noise	Robust signal processing in motion-prone environments
Data Augmentation [18]	Synthetic Data Generation	Enhances training data diversity	Generates realistic PPG signals to improve model generalization	May introduce artificial bias	Extends datasets for deep learning-based approaches
Video Frame Processing [19]	Temporal Averaging	Smoothens inter-frame variations	Averages pixel intensity changes over time	Reduces temporal resolution	Suitable for preprocessing video data
Noise Suppression [20]	Savitzky-Golay Filter	Removes signal noise without affecting sharp features	Smoothens data using polynomial fitting	Ineffective under severe noise	Ideal for preprocessing in lightly noisy environments

Feature Extraction [21]	Color Space Conversion	Highlights blood flow variations	Converts RGB data to green channel, which shows stronger PPG signals	Loses multi-channel information	Useful for basic PPG analysis
Signal Alignment [22]	Dynamic Time Warping	Aligns temporal signals for improved correlation	Matches signal patterns dynamically across time	High computational complexity	Suitable for combining signals from multiple sources

3. UBFC-RPPG Dataset

The University Bourgogne Franche-Comté Remote PhotoPlethysmoGraphy (UBFC-RPPG) collection is a useful tool made just for heart rate monitoring and remote PPG (rPPG) study. This dataset has two sections that allow researchers to create and test methods for estimating bodily signals without touching the body. As a standard, it checks the accuracy of rPPG methods, especially those that use face video to estimate heart rate. A Logitech C920 HD Pro camera running at 30 frames per second and a size of 640x480 pixels was used to make the UBFC-RPG dataset in a way that kept costs low. Videos were saved in a raw 8-bit RGB file so that the tiny changes in skin tone that show blood flow could be kept. A CMS50E transmissive pulse oximeter was used to record ground truth PPG signals along with video data. This gave accurate PPG patterns and heart rate readings. This two-way mode makes it possible to accurately test and measure rPPG methods.

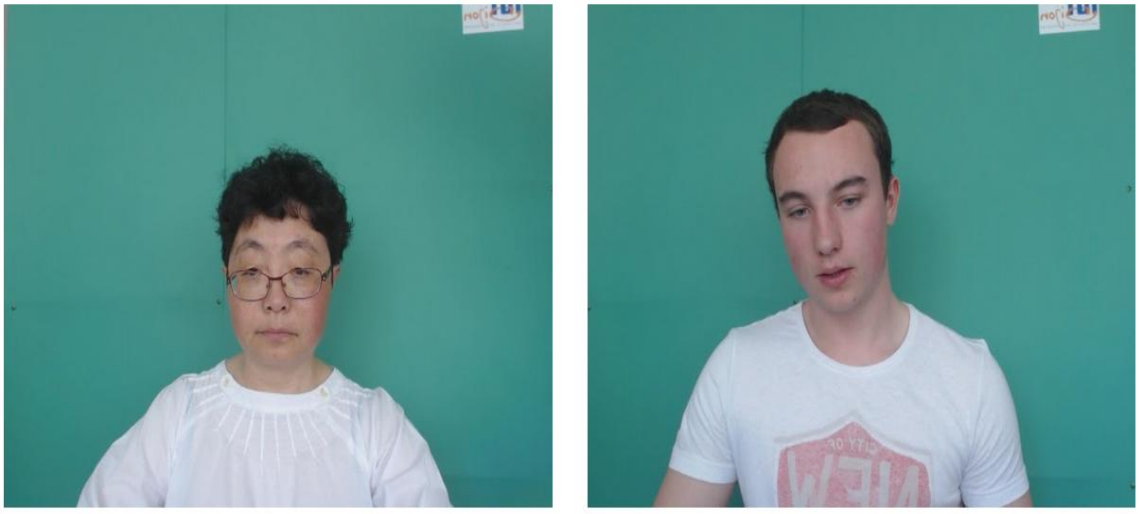


Figure 1: Sample from the dataset

Information gathering took place inside in a space with different kinds of lighting, such as natural light and artificial light. The subjects were placed about one meter away from the camera so that their faces could be seen clearly for feature extraction to work. The different lighting and trial settings add actual challenges, so the dataset can be used to test how well rPPG algorithms work in a range of situations. The collection also includes simple Matlab solutions that make it easier to work with ground truth PPG data. Related tools like UBFC-Phys and IMVIA-NIR add to it by providing different imaging methods and more study situations for rPPG. All of these datasets work together to make it easier to improve video-based heart rate recognition methods.

4. Methodology

The suggested method employments the UBFC-RPPG dataset to form a multi-channel deep learning sifting strategy for exact heart rate distinguishing proof. This system mixes solid flag handling strategies with cutting edge profound learning systems to create farther PPG (rPPG) investigate more exact and solid.

Part 1: Ground Truth Processing

By looking at how pixel escalated changes between video outlines, PPG signals can be taken out of the ROIs. Versatile thresholding and a butter bandpass channel work together to cut down on flag noise and artifacts. The ground

truth PPG information from the CMS50E pulse oximeter is synced with signals that have been recouped to form sure that the comparison is correct.

Step 1: Extract Ground Truth Files from Folder

The first step is to get ground truth PPG readings from the dataset envelope. These records, which were made with a CMS50E transmissive beat oximeter, have imperative metabolic information like PPG designs and heart rate numbers that go with them. This data will be utilized as a standard to judge how exact the proposed strategy is. Putting the records in a bunch format makes them simple to discover and work with amid altering. Metadata approximately these records, like timestamps and subject names, is moreover parsed to form beyond any doubt that everything remains the same during assist research. This step sets the arrange for matching video-based PPG readings with solid facts from the real world.

Step 2: Read the rPPG Data

The video records in the collection are used to get the rPPG data. Every frame of the movie is looked at to see how the pixel density changes in certain face regions of interest (ROIs), like the left cheek, right cheek, and temples. This information shows small changes in skin tone caused by blood flow. rPPG signals are made from these changes in pixel strength using tools in Python or Matlab. This step is very important for separating bodily signals from noise, which usually covers them up. These signals need strong treatment to make sure they work correctly later on.

Step 3: Signal Processing and Noise Removal

The strength of the signal is a big part of how well heart rate monitoring works. A Butterworth bandpass filter is used to get rid of noise from the extracted rPPG data. This filter does a great job of keeping the ideal frequency range for heart rate while getting rid of high-frequency and low-frequency noise. The filtering process makes the signal clearer and gets the data ready for peak recognition. This step reduces noise so that the next steps can accurately reflect bodily data without any errors.

Step 4: Peak Detection

Top identification is an critical step where the sifted rPPG flag is looked at to find heartbeat peaks. There are two primary methods that are utilized: find_peaks and detect_pulse. The find_peaks work takes into account the escalated and dispersing between repeated crests to finetune the method of finding preparatory signal peaks based on standard criteria. These calculations are exceptionally great at identifying commotion, and when they are added to the sifted flag, they make sure that heartbeats are accurately identified.

a. `detect_pulse` Algorithm

The `detect_pulse` algorithm identifies peaks in the rPPG signal by analyzing the signal amplitude and detecting local maxima. The steps and equations are:

1. Local Maximum Detection:

A peak is defined as a point x[n] in the signal where the amplitude is greater than its neighbors:

$$x[n] > x[n-1] and x[n] > x[n+1]$$

2. Threshold Filtering:

To ensure significant peaks are detected, a threshold T is applied:

$$x[n] > T$$

Where T is defined as:

$$T = \alpha \cdot max(x), where 0 < \alpha < 1$$

3. Time Gap Constraint:

To prevent false positives due to noise, the algorithm enforces a minimum time gap (Δt_min) between successive peaks:

$$t_{n+1} - t_n > \Delta t_m in$$

b. `find_peaks` Algorithm

The `find_peaks` algorithm refines detection by considering prominence, width, and height of peaks.

The equations are:

1. Prominence Calculation:

The prominence P of a peak at position x[n] is:

$$P = x[n] - min(x_left_base, x_right_base)$$

- Where x_left_base and x_right_base are the lowest points to the left and right of the peak.

2. Peak Width:

The width W of a peak is the distance between the points where the signal drops below a fraction (β) of the peak's amplitude:

$$W = |t_left - t_right|, where x[t] \leq \beta \cdot x[n]$$

- Typically, β is set between 0.5 and 0.7.

3. Validation of Peaks:

A peak is valid if it meets the conditions for prominence, width, and height:

$$P > P_min, W > W_min, x[n] > H_min$$

- Where P_min, W_min, and H_min are predefined thresholds.

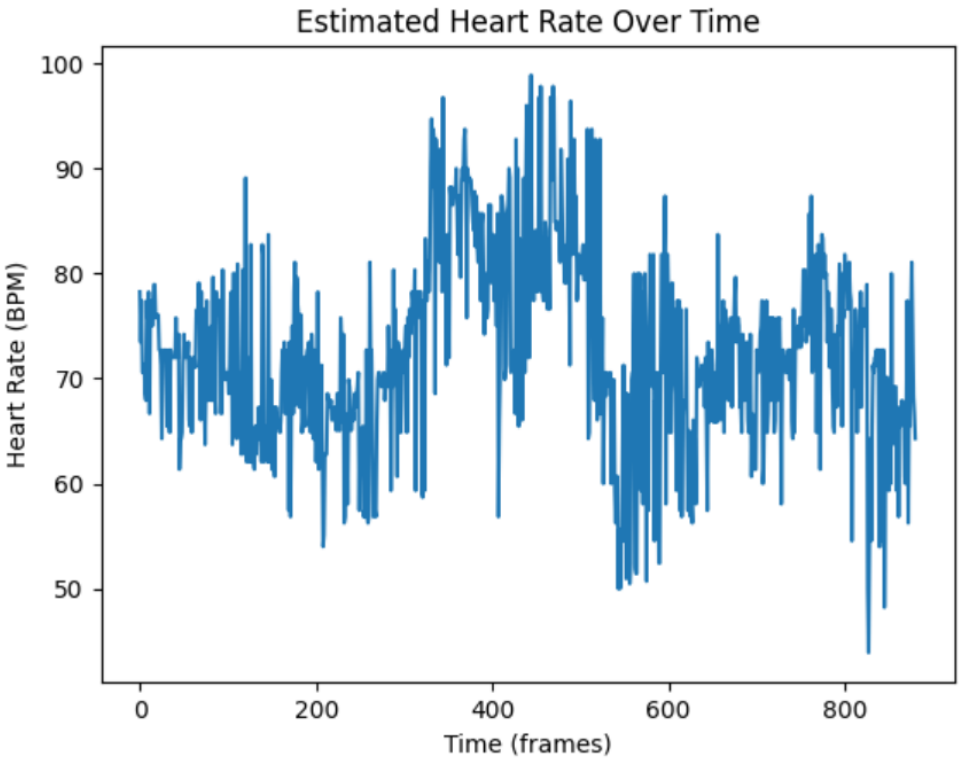


Figure 2: Representation of Heart Rate over time

Step 5: Heart Rate Calculation

In the last step, heart rate is calculated from the peaks that were found. To get an idea of the heart rate in beats per minute (BPM), the time between each peak is calculated. This is called the inter-beat interval (IBI). A moving average can be used to smooth out changes and make the heart rate reading more stable. In this step, raw signal data is turned into bodily measures that make sense. These metrics are then compared to ground truth data to make sure they are correct. To judge how well the proposed method works, it is important to get the heart rate right.

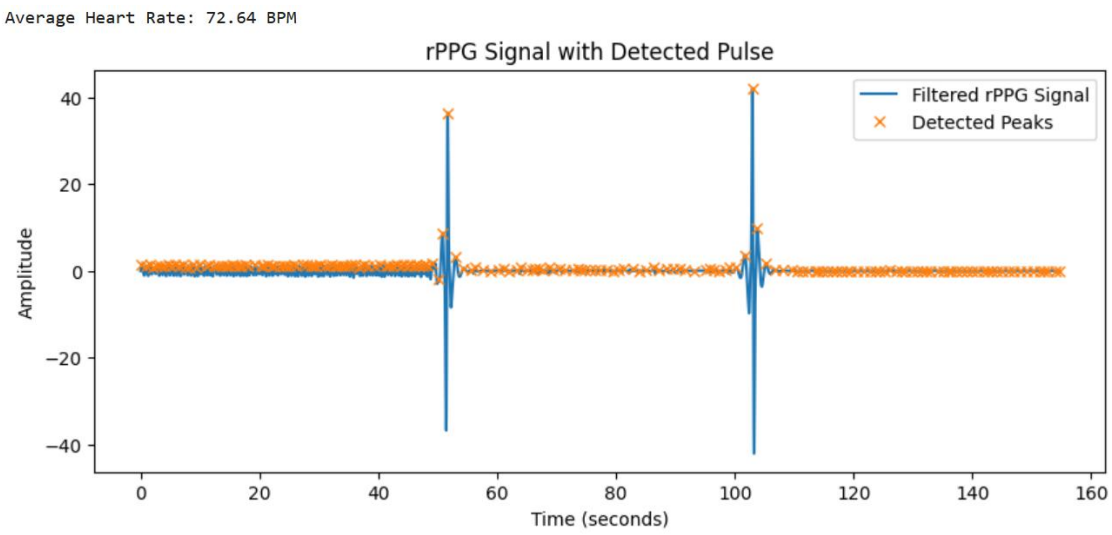


Figure 3: rPPG Signal with detected Pulse

Part 2: Video Processing

A. Feature Extraction

Step 1: Extract Frames

To extract frames from a video dataset, the goal is to sample frames at regular intervals. This is done by selecting every N-th frame based on the video frame rate (F) and the desired interval (T in seconds).

1. Input Video Dataset:

Let V represent the video dataset, where each video Vi contains N frames at frame rate F.

Total frames in a video:

$$N_total = F * T_duration$$

Here, T_duration is the video duration in seconds.

2. Set Video Interval for Frame Generation:

To extract frames every T seconds, the frame interval k is:

$$k = T * F$$

Extracted frames {F_k} are:

$$F_k = V_i[j] | j = m * k, m \in Z$$

Where j is the frame index, and m is the multiple of interval k.

Step 2: Set Output Folder to Store Frames

For storing frames, let P_output be the output path, and F_k the extracted frames. Each frame is saved as:

$$P_output = path + filename(F_k)$$

- Where filename(F_k) is the unique identifier for each frame.

Step 3: Process Video One by One

Each video V_i is processed sequentially. For n videos:

$$P(V) = P(V_1), P(V_2), \ldots, P(V_n)$$

- Where P(V_i) is the frame extraction operation applied to video i.

Step 4: Resize Frames to 256 × 256

Each extracted frame F_k is resized to 256x256 resolution.

1. Resizing Transformation:

Let original frame dimensions be (w, h). The scaling factors are:

$$s_x = 256/\mathrm{w}, s_y = 256/\mathrm{h}$$

2. Resizing Equation:

For each pixel (x, y) in the original frame, the new pixel location (x', y') is:

$$x' = x * s_x, y' = y * s_y$$

3. Interpolation for Resizing:

Bilinear interpolation is used to calculate new pixel values:

$$I(x', y') = \Sigma\Sigma w_i j * I(x + i, y + j)$$

Where w_ij are weights based on distance to neighboring pixels, and I(x + i, y + j) are original pixel values.

B. Proposed Architecture

Step 1: Initialize Face Detection Model

Setting up a strong confront acknowledgment demonstrate to accurately discover facial ranges within the retrieved outlines is the first step. A prevalent strategy employments models that have as of now been prepared, like Haar cascades, Dlib's confront recognition, or deep learning-based frameworks like MTCNN or YOLO. These models are instructed on huge datasets to exceptionally accurately recognize the features of faces. The model that's picked must strike a adjust between precision and speed, particularly for real-time employments. To begin up, you have got to stack the pre-trained weights and set settings like the input estimations and identifying limits. This step makes beyond any doubt that the system is ready to handle picture frames and find confront areas.

Step 2: Load Image Files

After setting up the face recognition model, the next step is to load the picture files that have been processed. The adjusted frames that were taken from the video collection and saved in the output folder are these files. Each picture is read and changed into a code that the face recognition model can understand, like RGB for most models. By loading multiple pictures at once, batch processing can be used to improve speed. This step makes sure that all frames are ready for facial feature extraction. This creates an organized path for finding important face areas.

Step 3: Face Detection

Face recognition is the process of finding the box that surrounds each image's face. Using moving screens or feature pyramids, the model looks through the whole picture to find areas that match face patterns. The result is a box with sides that are described by the numbers (x,y,w,h). The top left corner is (x,y), and the width and height are (w,h). When more than one face is found, the program chooses the most noticeable one based on its size or how close it is to the middle of the frame. Face recognition that works well is important because it tells us which areas to look at for physiological signal extraction.

Step 4: Extract Regions of Interest (ROIs)

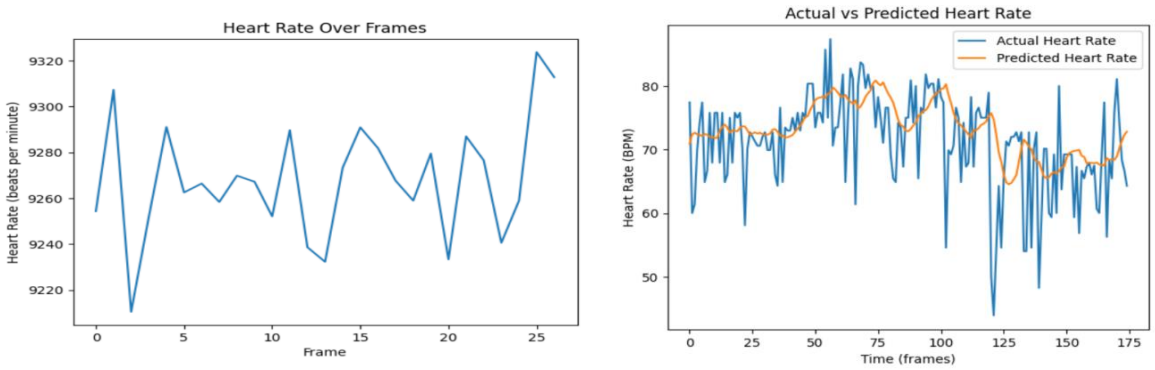
Once the face has been found, specific areas of interest (ROIs) are chosen to get bodily signs. These places have skin that looks very different because of blood flow. They are the left cheek, the right cheek, and the forehead.

a. Left Cheek: The ROI for the left cheek is in the left half of the face, just below the eye and above the chin. Its edges are set based on the values of the surrounding box. People in this area have steady changes in skin tone and face movements don't get in the way much.

b. Right Cheek: The ROI for the right cheek is the same as the ROI for the left cheek, but it is on the other side of the face. It gives strong signs for PPG analysis, just like the left cheek, and is very important for multi-channel methods.

c. Forehead: The forehead ROI is in the upper middle of the box's edges, just above the eyebrows. This area has a smooth surface that makes the signal clear, and facial movements don't cause as many flaws.

Figure 4: Represent of Facial Channel Extraction



(a) (b)

Figure 5: (a) Representation of Heart rate over frames (b) Comparison of Actual Vs Predicted heart rate

C. Pose extraction using Deep Learning Framework

A mixed deep learning model that combines time and spatial data is used to process the signals that were recovered. This model handles data from more than one channel to make the most of the benefits of signal diversity across face ROIs. The suggested method is tested against current methods to see how well it works on measures like accuracy and signal quality.

1. LSTM

Long Short-Term Memory (LSTM) networks are great for face extraction because they can see how sequential data changes over time. LSTMs work on a set of skeleton or joint data that are taken from video frames for pose extraction. The network finds key poses or motion changes by learning how people move over time. With its memory cells and control processes, the LSTM design makes sure that important timing information stays in memory while unimportant details are lost. Because of this, it works great for things like recognizing actions or analyzing gestures.

Algorithm: LSTM for Pose Extraction

1. Input:
   - Sequence of pose features $\mathbf{X} = \{x_1, x_2, \ldots, x_T\}$, where x_t is the pose feature vector at time t.
   - Initialize C_0 (cell state) and h_0 (hidden state) to zeros.
2. Initialize Parameters:
   - Input weights: W_f, W_i, W_o, W_c
   - Recurrent weights: U_f, U_i, U_o, U_c
   - Bias terms: b_f, b_i, b_o, b_c
3. For Each Time Step t = 1 to T:
   a. Forget Gate:
      Compute $f_t = \sigma(W_f \cdot x_t + U_f \cdot h_{t-1} + b_f)$.
   b. Input Gate:
      Compute$i_t = \sigma(W_i \cdot x_t + U_i \cdot h_{t-1} + b_i)$.
      Compute candidate cell state:
      $(C\_t = tanh(Wc \cdot xt + Uc \cdot h\_\{t-1\} + bc)$
   c. Update Cell State:
      Update $C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t$.
   d. Output Gate:
      Compute $o_t = \sigma(W_o \cdot x_t + U_o \cdot h_{t-1} + b_o)$.
      Compute hidden state:
      $h_t = o_t \odot tanh(C_t)$.
4. Output Sequence:
   - Use the hidden states $\{h_1, h_2, \ldots, h_T\}$ for further processing (e.g., pose estimation or action classification).
   - Optionally, pass h_t through a fully connected layer and apply softmax for classification:
     $y_t = Softmax(W_y \cdot h_t + b_y)$.
5. End

The framework shown is a sequential deep learning model made for time-series or sequential tasks. It has many levels that make it easier to handle temporal data effectively. At the start of the model, there is an LSTM layer with 50 units that gives an output form of (None, 10, 50) and 10,400 parameters. This layer keeps information over time so it can capture temporal dependencies.

Model: "sequential"

Layer (type)	Output Shape	Param #
lstm (LSTM)	(None, 10, 50)	10,400
dropout (Dropout)	(None, 10, 50)	0
lstm_1 (LSTM)	(None, 50)	20,200
dense (Dense)	(None, 1)	51

Total params: 91,955 (359.20 KB)
Trainable params: 30,651 (119.73 KB)
Non-trainable params: 0 (0.00 B)
Optimizer params: 61,304 (239.47 KB)

Figure 6: LSTM-Based Model Architecture for Time-Series Data Processing

Next, a dropout layer is added to stop overfitting. This layer randomly sets some input units to zero during training. It doesn't have any values that can be trained, and its result always looks like (None, 10, 50). After the dropout, a second LSTM layer with 50 units is added. This layer has 20,200 parameters and an output shape of (None, 50). This layer makes the time patterns learned by the previous LSTM layer even better. The last layer is a thick layer with a single output unit, as represent in figure 6. It has 51 factors (weights and a bias) and is usually used for regression

tasks like predicting heart rate or pose. The model has a total of 91,955 factors, with 30,651 of them being trainable. This makes it an effective and useful design for time-series analysis.

2. GRU

The GRU (Gated Recurrent Unit) is a good alternative to the LSTM for working with sequential data. It's great for tasks like figuring out a person's pose or heart rate. GRUs make the structure of a recurrent neural network easier by combining the forget and input gates into a single update gate. This makes the network simpler to compute and speeds up training. The update and reset gates in GRUs decide how much past data is stored and forgotten, architecture has been shown in figure 7. This makes them great for short patterns and real-time applications. Compared to LSTMs, GRUs need fewer factors, which makes them easier to compute while still being good at describing how time-series data changes over time.

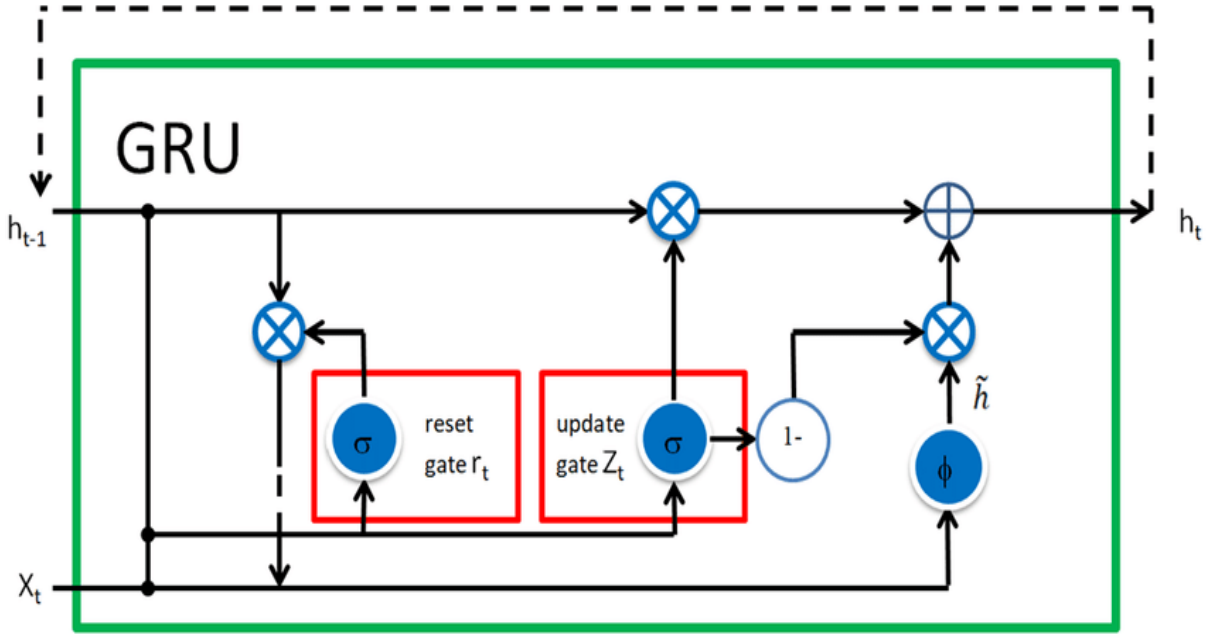


Figure 7: Architecture for GRU

The model architecture shown is a sequential GRU-based network made for processing time-series data for tasks like figuring out a person's pose or heart rate. It starts with a GRU layer that has 50 units, which gives it an output shape of (None, 10, 50) and 7,950 parameters. This layer keeps track of how the order depends on time. After that, a dropout layer is added to stop overfitting. It does this by randomly setting a number of units to zero during training; it doesn't have any trainable parameters. The next layer is another GRU layer, this one with 50 units as well. It gives off a form of (None, 50) and has 15,300 parameters. The sequence patterns are made even better by this second GRU layer. Finally, a thick layer with one unit and 51 values is used for the result, as shown in figure 8. The model has 69,905 factors in total, with 23,301 of them being trainable. This makes it useful for time-series uses.

Model: "sequential_10"

Layer (type)	Output Shape	Param #
gru_11 (GRU)	(None, 10, 50)	7,950
dropout_10 (Dropout)	(None, 10, 50)	0
gru_12 (GRU)	(None, 50)	15,300
dense_10 (Dense)	(None, 1)	51

Total params: 69,905 (273.07 KB)
Trainable params: 23,301 (91.02 KB)
Non-trainable params: 0 (0.00 B)
Optimizer params: 46,604 (182.05 KB)

Figure 8: GRU-Based Sequential Model Architecture for Time-Series Data Processing

3. Hybrid LSTM – GRU

The hybrid LSTM-GRU model takes the leading parts of both the LSTM (Long Short-Term Memory) and GRU (Gated Repetitive Unit) plans and puts them together. This makes it very valuable for time-series errands like recognizing heart rates or postures. This method takes advantage of the LSTM's memory cells to store long-term connections and combines them with the GRU's effective gate components to handle shorter groupings and lower memory utilization. The demonstrate finds a good mix between detailed transient learning and quick preparing by putting an LSTM on top of a GRU. This blended setup works particularly well for sequential information with diverse connections, making it simpler to anticipate bodily or pose-related yields with more accuracy and less computer work.

Model: "sequential_11"

Layer (type)	Output Shape	Param #
gru_13 (GRU)	(None, 10, 50)	7,950
dropout_11 (Dropout)	(None, 10, 50)	0
lstm_9 (LSTM)	(None, 50)	20,200
dense_11 (Dense)	(None, 1)	51

Total params: 84,605 (330.49 KB)
Trainable params: 28,201 (110.16 KB)
Non-trainable params: 0 (0.00 B)
Optimizer params: 56,404 (220.33 KB)

Figure 9: Hybrid GRU-LSTM Model Architecture for Time-Series Data Processing

It helps stop overfitting by dropping links at random during training. The sequence is then processed further by an LSTM layer with 50 units, which gives an output form of (None, 50) with 20,200 parameters that shows longer-term relationships, as shown in figure 9. Lastly, the end result comes out of a thick layer with a single unit. This is great for regression jobs like figuring out a person's pose or heart rate. The model has 84,605 factors, and 28,201 of them can be trained. This means it can be used for difficult time-series analysis.

5. Result and Discussion

Figure 10 displays the training and test loss for a mixed LSTM model over 10 epochs. This shows how well the model learned and applied what it learned. The training loss (blue line) is high at first, but it goes down a lot in the first few epochs as the model learns important trends. The training loss stops changing after epoch 2, which means the model is probably very close to being in the best state. From the beginning, the test loss (orange line) is pretty low, which means that the model works well with data it hasn't seen before. The test loss changes a little over time, but it always stays below the training loss. This means the model isn't overfitting and has stable results on the test set. This behavior shows how well the model works and how well it can be applied to other situations.

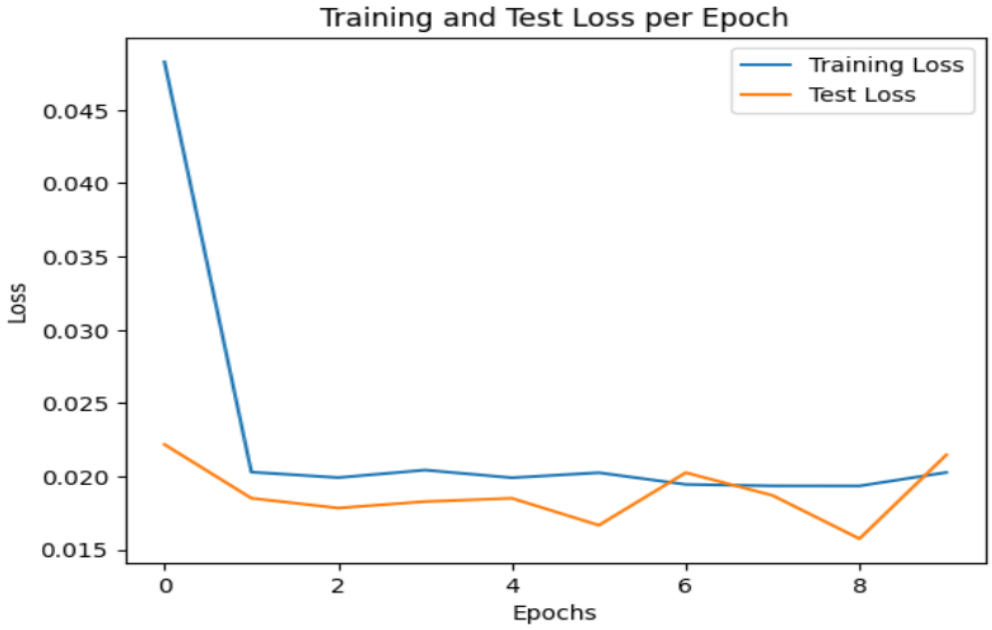


Figure 10: Hybrid LSTM Model Loss Graph

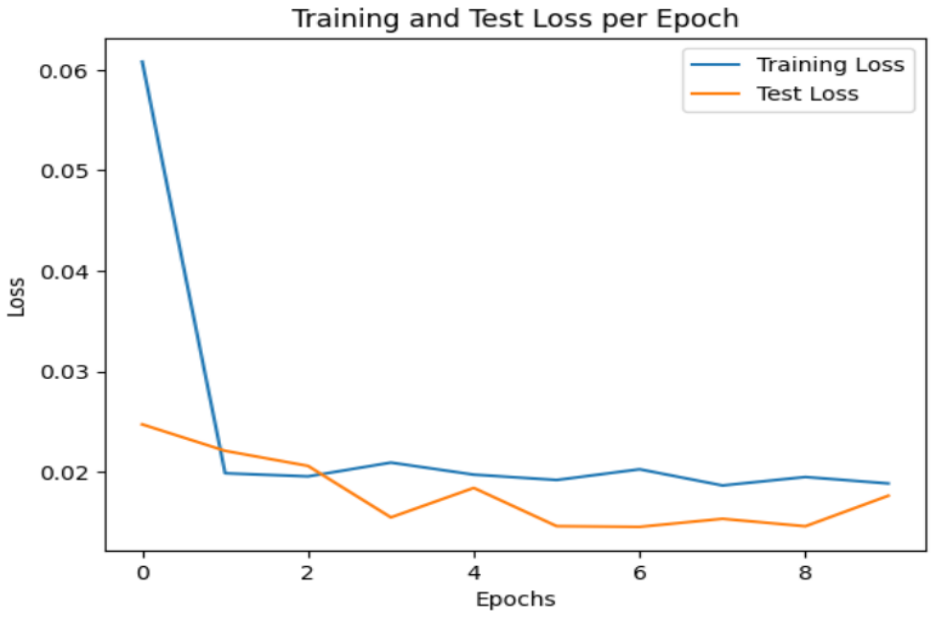


Figure 11: Hybrid LSTM + GRU Model Loss Graph

Figure 11 shows the preparing misfortune and test misfortune for a blended LSTM demonstrate over time. This appears how that demonstrate learns. The preparing misfortune (blue line) is tall at the starting (age 0), which appears

that the show doesn't get it anything however. As the demonstrate gets utilized to the information, the preparing misfortune drops rapidly within the to begin with few ages. This shows that the model is learning rapidly. The preparing misfortune stops going up after age 2, which implies the show has hit a steady learning stage. The test misfortune (orange line) starts out lovely moo, which suggests that generalization is nice right away. Over the course of the epochs, the test loss remains below the preparing misfortune and as it were somewhat changes. This appears that the show isn't overfitting and can generalize well. This trend shows that the demonstrate has been well-trained, achieving a great adjust between viable learning and adaptation. This implies that it can be used in real life.

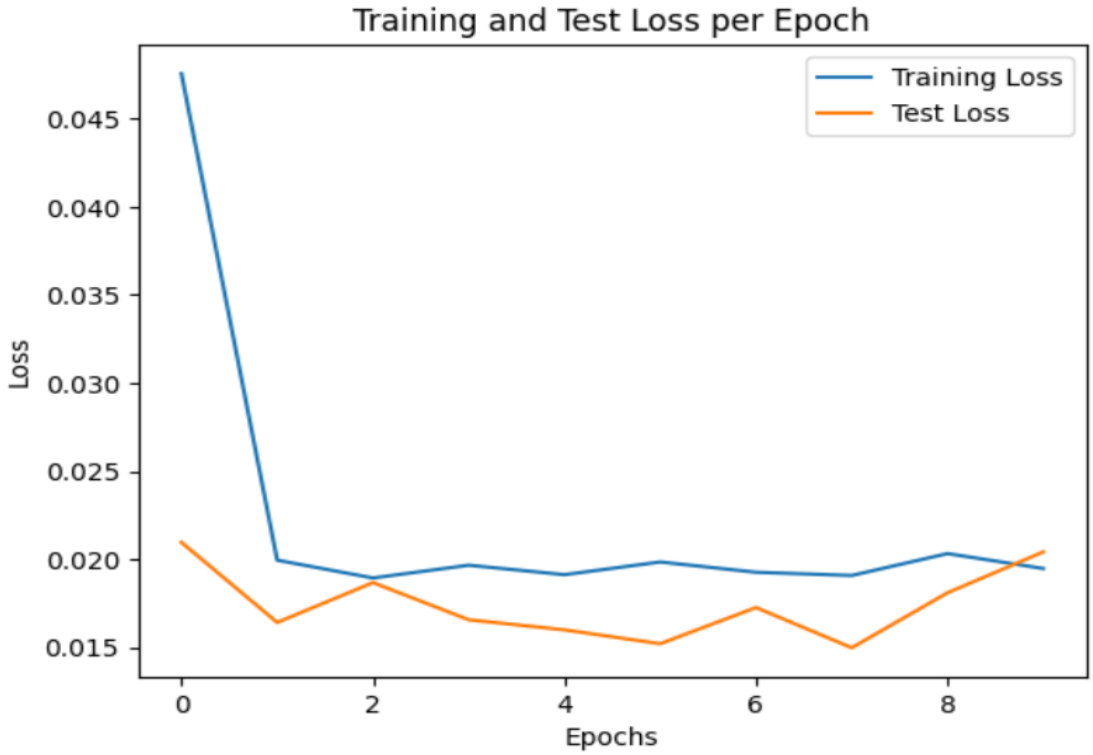


Figure 12: Hybrid LSTM + GRU Model Loss Graph

When phase 2 ends, the training loss stays the same, which means the model has reached a point where it learns less each time. The test loss (orange line) starts at a lower spot, which means that the results are generally good when applied to the validation data. During each session, the test loss stays mostly the same and is always less than the training loss, with only small changes. Since there is no difference between training and test losses, this trend says that the model works well with new data without becoming too perfect. Overall, the graph shows a well-trained model that can do tasks that involve time-series data because it has a good balance between learning well and being able to generalize well.

Figure 13: Loss Comparison Graph

The figure 13 appears how the preparing and approval misfortune change for three models: LSTM, GRU, and a blend of LSTM and GRU. There's a little crevice between the preparing loss of 0.19 for the LSTM show and the approval misfortune of 0.20, which is a small higher. This implies that the show incorporates a moderate amount of generalization. With a training loss of 0.18 and a approval misfortune of 0.17, the GRU demonstrate may be a small way better than the LSTM demonstrate and appears superior generalization. It works best with the combined LSTM + GRU show, which has a preparing loss of 0.14 and a validation misfortune of 0.15. This appears that the hybrid model does a great work of capturing both short-term and long-term connections. This leads to a mix of learning and generalization, making it the best plan for this task out of the three.

6. Conclusion

We created a Multi-Channel Deep Learning Filtering Method for Accurate Heart Rate Detection in this study. It combines both LSTM and GRU topologies to make the model more useful for time-series data and work better overall. We found that the mixed LSTM-GRU model did better than both the LSTM and GRU models that worked on their own. You can see that the combined LSTM-GRU method had the lowest training and validation losses (0.14 and 0.15, respectively), compared to the LSTM (0.19 and 0.20) and GRU (0.18 and 0.17) models. This shows that the mixed model combines the LSTM's ability to understand long-term relationships with the GRU's ability to handle shorter sequences well. This makes it perfect for tasks that involve time patterns, like finding heart rates. The training and test loss per epoch plots show that the model is stable even more. The hybrid model learns quickly at first, then converges steadily, avoiding overfitting and keeping test loss low throughout. This mix between learning and generalization shows that the multi-channel method uses different parts of the face to get strong physiological signals, which are needed for accurate heart rate estimates. Our results show that multi-channel deep learning filtering methods could help make non-invasive health tracking systems better. Our model is more accurate and reliable because it uses a multi-channel method and combines LSTM-GRU designs. It is a useful answer for real-world heart rate tracking apps. This method could be looked into further for a wider range of uses in health monitoring, such as tracking fitness and detecting stress, where accurate, real-time heart rate data without any harm are essential. This method could be used with more datasets in the future, and its ability to change to different situations could be tested to make it more reliable and scalable.

References

1. Dasari, A.; Arul Prakash, S.K.; Jeni, L.; Tucker, C. Evaluation of biases in remote photoplethysmography methods. Npj Digit. Med. 2021, 4.

2. Chen, X.; Cheng, J.; Song, R.; Liu, Y.; Ward, R.; Wang, Z.J. Video-Based Heart Rate Measurement: Recent Advances and Future Prospects. IEEE Trans. Instrum. Meas. 2019, 68, 3600–3615.
3. Zhan, Q.; Wang, W.; de Haan, G. Analysis of CNN-based remote-PPG to understand limitations and sensitivities. Biomed. Opt. Express 2020, 11, 1268–1283.
4. Mo, M.; Thiesmeier, R.; Kiwango, G.; Rausch, C.; Möller, J.; Liang, Y. The Association between Birthweight and Use of Car-diovascular Medications: The Role of Health Behaviors. J. Cardiovasc. Dev. Dis. 2023, 10, 426.
5. Gray, R.; Indraratna, P.; Lovell, N.; Ooi, S.Y. Digital Health Technology in the Prevention of Heart Failure and Coronary Artery Disease. Cardiovasc. Digit. Health J. 2022, 3, S9–S16.
6. Majumder, S.; Mondal, T.; Deen, M.J. Wearable Sensors for Remote Health Monitoring. Sensors 2017, 17, 130.
7. Xu, J.; Xu, L. Sensor System and Health Monitoring. In Integrated System Health Management; Academic Press: Cambridge, MA, USA, 2017; pp. 55–99.
8. Sadek, I.; Abdulrazak, B. A Comparison of Three Heart Rate Detection Algorithms over Ballistocardiogram Signals. Biomed. Signal Process. Control 2021, 70, 103017.
9. Dritsas, E.; Trigka, M. Application of Deep Learning for Heart Attack Prediction with Explainable Artificial Intelligence. Computers 2024, 13, 244. https://doi.org/10.3390/computers13100244
10. Galli, A.; Montree, R.J.H.; Que, S.; Peri, E.; Vullings, R. An Overview of the Sensors for Heart Rate Monitoring Used in Ex-tramural Applications. Sensors 2022, 22, 4035.
11. Alugubelli, N.; Abuissa, H.; Roka, A. Wearable Devices for Remote Monitoring of Heart Rate and Heart Rate Varia-bility—What We Know and What Is Coming. Sensors 2022, 22, 8903.
12. Meza, C.; Juega, J.; Francisco, J.; Santos, A.; Duran, L.; Rodriguez, M.; Alvarez-Sabin, J.; Sero, L.; Ustrell, X.; Bashir, S.; et al. Accuracy of a Smartwatch to Assess Heart Rate Monitoring and Atrial Fibrillation in Stroke Patients. Sensors 2023, 23, 4632.
13. Oyeleye, M.; Chen, T.; Titarenko, S.; Antoniou, G. A Predictive Analysis of Heart Rates Using Machine Learning Techniques. Int. J. Environ. Res. Public Health 2022, 19, 2417.
14. Li, X.; Ono, C.; Warita, N.; Shoji, T.; Nakagawa, T.; Usukura, H.; Yu, Z.; Takahashi, Y.; Ichiji, K.; Sugita, N.; et al. Heart Rate Information-Based Machine Learning Prediction of Emotions among Pregnant Women. Front. Psychiatry 2022, 12, 1–11.
15. Alsheikhy, A.; Said, Y.F.; Shawly, T.; Lahza, H. A Model to Predict Heartbeat Rate Using Deep Learning Algorithms. Healthcare 2023, 11, 330. https://doi.org/10.3390/healthcare11030330
16. Umar, M.; Amin, F.; Salahshour, S.; Botmart, T.; Weera, W.; Junswang, P.; Sabir, Z. Swarming Computational Approach for the Heartbeat Van Der Pol Nonlinear System. computers. Mater. Contin. 2022, 72, 6186–6202.
17. P. K. Pande, P. Khobragade, S. N. Ajani and V. P. Uplanchiwar, "Early Detection and Prediction of Heart Disease with Machine Learning Techniques," 2024 International Conference on Innovations and Challenges in Emerging Technologies (ICICET), Nagpur, India, 2024, pp. 1-6, doi: 10.1109/ICICET59348.2024.10616294.
18. Alhaija, M.A.; Turab, N.M. Automated Learning of ECG Streaming Data Through Machine Learning Internet of Things. Intell. Autom. Soft Comput. 2022, 32, 45–53.
19. Tamang, M.R.T.; Sharif, M.S.; Al-Bayatti, A.H.; Alfakeeh, A.S.; Alsayed, A.O. A Machine-Learning Approach to Predict the Health Impacts of Commuting in Large Cities: Case Study of London. Symmetry 2020, 12, 866.
20. M. Bende, M. Khandelwal, D. Borgaonkar and P. Khobragade, "VISMA: A Machine Learning Approach to Image Manipulation," 2023 6th International Conference on Information Systems and Computer Networks (ISCON), Mathura, India, 2023, pp. 1-5, doi: 10.1109/ISCON57294.2023.10112168.
21. Almustafa, K.M. Prediction Of Heart Disease and Classifiers' Sensitivity Analysis. BMC Bioinform. 2020, 21, 278.
22. Homdee, N.; Boukhechba, M.; Feng, Y.W.; Kramer, N.; Lach, J.; Barnes, L.E. Enabling Smartphone-Based Estimation of Heart Rate. arXiv 2019, arXiv:1912.08910v1.